Joint Inventors

Docket No. INTEL/17582
P17582

"EXPRESS MAIL" mailing label No.
EL 995292456 US
Date of Deposit: **September 30, 2003**

I hereby certify that this paper (or fee) is being deposited with the United States Postal Service "EXPRESS MAIL POST OFFICE TO ADDRESSEE" service under 37 CFR §1.10 on the date indicated above and is addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450

Magda Greer

Magda Greer

APPLICATION FOR
UNITED STATES LETTERS PATENT

SPECIFICATION

TO ALL WHOM IT MAY CONCERN:

Be it known that We, Michael A. Rothman, a citizen of United States of America, residing at 3311 11th Ave. Ct. NW, Gig Harbor, Washington 98335; Vincent J. Zimmer, a citizen of United States of America, residing at 1937 South 369th Street, Federal Way, Washington 98003 and Harry L. Hsiung, a citizen of United States of America, residing at 5512 Crestview Loop NE, Olympia, Washington 98516 have invented a new and useful **"METHODS AND APPARATUS TO PROVIDE CONDITIONAL LEGACY SUPPORT"**, of which the following is a specification.

METHODS AND APPARATUS TO PROVIDE CONDITIONAL LEGACY SUPPORT

FIELD OF THE DISCLOSURE

[0001] The present disclosure relates generally to processor systems and, more particularly, to methods and apparatus to provide conditional legacy support within processor systems.

BACKGROUND

[0002] Computer systems and computer software are constantly evolving. As computer systems evolve, the computer software designed to run on them must also evolve. Often, systems and software evolve at a constant pace, and a new computer system design can execute the latest computer software design. However, there are instances in which a new computer system is designed to operate old computer, software such as an old operating system. For this reason, new computer designs must maintain backward compatibility with old computer software.

[0003] It is known that computer system designers implement a firmware resource on computer system designs that, in a pre-boot environment, initializes a computer system. This firmware resource is known as a basic input output system (BIOS) and is used to provide basic functionality (e.g., display output, keyboard input, disk drive operation, etc.) to a computer system prior to executing an operating system boot process. The BIOS is generally stored on a non-volatile memory device such as a ROM or a flash memory that is designed or integrated into most computer systems.

[0004] In recent years, newer firmware resources have been designed that provide functionality similar to the BIOS. One such resource is known as an extensible firmware interface (EFI) that initializes a computer system prior to execution of an

operating system boot process. In general, the EFI is targeted toward supporting newer operating systems and application programs. However, for new computer designs to maintain backward compatibility with older or legacy software, the EFI can provide legacy support through compatibility resources known as compatibility support modules. The compatibility support modules are firmware instruction modules that, upon execution, enable a computer system to provide support for legacy computer software, such as legacy operating systems. The compatibility support modules are stored in the ROM or flash memory on which the EFI is stored so that the EFI can unconditionally load the compatibility support module to the computer system prior to executing a software boot process.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] FIG. 1 is a diagram of an example processor system.

[0006] FIG. 2 is a known memory allocation configuration of a pre-boot initialization firmware image.

[0007] FIG. 3 is a flow diagram of a known initialization process that initializes a processor system prior to executing an operating system boot process.

[0008] FIG. 4 is an example memory allocation configuration that supports a cross-memory implementation.

[0009] FIG. 5 is a flow diagram of an example initialization process that initializes a processor system prior to executing an operating system boot process.

DETAILED DESCRIPTION

[0010] Although the following discloses example systems including, among other components, software or firmware executed on hardware, it should be noted that such

systems are merely illustrative and should not be considered as limiting. For example, it is contemplated that any or all of these hardware and software components could be embodied exclusively in hardware, exclusively in software, exclusively in firmware or in some combination of hardware, firmware and/or software. Accordingly, while the following describes example systems, persons of ordinary skill in the art will readily appreciate that the examples are not the only way to implement such systems.

[0011] Turning now to FIG. 1, an example processor system 100 includes a processor 102 having associated system memory 104. The system memory 104 may include one or more of a random access memory (RAM) 106, a read only memory (ROM) 108 and a flash memory 110. The ROM 108 and the flash memory 110 of the illustrated example may respectively include boot blocks 109 and 112.

[0012] The processor 102, in the example of FIG. 1, is coupled to an interface, such as a bus 114 to which other peripherals or devices are interfaced. In the illustrated example, the peripherals interfaced to the bus 114 include an input device 116, a disk controller 120 communicatively coupled to a mass storage device 122 (i.e., hard disk drive) having a host protected area 124, and a removable storage device drive 126. The removable storage device drive 126 may include associated removable storage media 128, such as magnetic or optical media.

[0013] The example processor system 100 of FIG. 1 also includes an adapter card 130, which is a peripheral coupled to the bus 114 and further coupled to a display device 132.

[0014] The example processor system 100 may be, for example, a conventional desktop personal computer, a notebook computer, a workstation or any other computing device. The processor 102 may be any type of processing unit, such as a

microprocessor from the Intel® Pentium® family of microprocessors, the Intel® Itanium® family of microprocessors, and/or the Intel XScale® family of processors.

[0015] The memories 106, 108, and 110, which form some or all of the system memory 104, may be any suitable memory devices and may be sized to fit the storage demands of the system 100. The ROM 108, the flash memory 110, and the mass storage device 122 are non-volatile memories. Additionally, the mass storage device 122 may be, for example, any magnetic or optical media that is readable by the processor 102.

[0016] The input device 116 may be implemented by a keyboard, a mouse, a touch screen, a track pad or any other device that enables a user to provide information to the processor 102.

[0017] The display device 132 may be, for example, a liquid crystal display (LCD) monitor, a cathode ray tube (CRT) monitor, or any other suitable device that acts as an interface between the processor 102 and a user via the adapter card 130. The adapter card 130 is any device used to interface the display device 132 to the bus 114. Such cards are presently commercially available from, for example, Creative Labs and other like vendors.

[0018] The removable storage device drive 126 may be, for example, an optical drive, such as a compact disk-recordable (CD-R) drive, a compact disk-rewritable (CD-RW) drive, a digital versatile disk (DVD) drive or any other optical drive. It may alternatively be, for example, a magnetic media drive. The removable storage media 128 is complimentary to the removable storage device drive 126, inasmuch as the media 128 is selected to operate with the drive 126. For example, if the removable storage device drive 126 is an optical drive, the removable storage media 128 may be a CD-R disk, a CD-RW disk, a DVD disk or any other suitable optical

disk. On the other hand, if the removable storage device drive 126 is a magnetic media device, the removable storage media 128 may be, for example, a diskette, or any other suitable magnetic storage media.

[0019] The example processor system 100 also includes a network adapter 136 (i.e., a processor peripheral), such as, for example, an Ethernet card or any other card that may be wired or wireless. The network adapter 136 provides network connectivity between the processor 102 and a network 140, which may be a local area network (LAN), a wide area network (WAN), the Internet, or any other suitable network. As shown in FIG. 1, further processor systems 144 may be coupled to the network 140, thereby providing for information exchange between the processor 102 and the processors of the processor systems 144.

[0020] FIG. 2 is a known memory allocation configuration of a processor system initialization firmware image 202. The firmware image 202 enables configuration of a processor and a processor system (i.e., the processor 102 and the processor system 100 of FIG. 1). The firmware image 202 includes processor boot instructions 204, main firmware instructions 206, a compatibility support module 208, and a user/hardware variable space 210, all of which are stored on a single non-volatile memory coupled to a processor (i.e., the processor 102 of FIG. 1) for accessing prior to loading and executing a software boot target (e.g., an operating system or application program). As shown in FIG. 2, the firmware image 202 may be completely stored on the flash memory 110 or alternatively on any other single non-volatile memory such as the ROM 108 of FIG. 1.

[0021] During execution, the processor boot instructions 204 cause the processor 102 to configure basic processor resources such as initializing system clocks, initializing memory interfaces, zeroing registers, resetting counters and the like,

thereby enabling the processor 102 to further initialize portions of the processor system 100. Furthermore, the processor boot instructions 204 inform the processor 102 of the location of the main firmware instructions 206.

[0022] In general, the main firmware instructions 206 may be associated with a processor system initialization resource known as an extensible firmware interface (EFI) that is comprised of instructions associated with initializing portions of the processor system 100. The responsibilities of the EFI 206 are similar to those of a traditional or legacy basic input output system known as a BIOS in that both are configured to establish a hardware/software interface prior to an operating system boot process and cause the processor system 100 to load and execute an operating system boot target. The EFI 206 is generally configured to support legacy and future operating systems and application programs, while the BIOS is limited to supporting legacy operating systems and application programs. Accordingly, the EFI 206 may replace the BIOS. The EFI 206 is generally designed to support future operating systems and application programs in a native manner, while providing support for legacy operating systems and application programs through firmware modules known as compatibility support modules.

[0023] As shown in FIG. 2, a compatibility support module 208 may also be stored in the flash memory 110. The compatibility support module 208 includes instructions that may be executed by the processor 102 to provide support for legacy operating systems or application programs by configuring the processor system 100 to operate as if the legacy BIOS were present. In particular, the compatibility support module 208 initializes a BIOS Data Area and legacy interrupt vectors similar to the manner in which the BIOS does.

[0024] A user/hardware variable space 210 is also included in the flash memory 110. The user/hardware variable space 210 stores initialization parameters specific to the processor system 100 such as processor type, mass storage device type, memory configuration settings, etc. The initialization parameters are used by the EFI 206 and the compatibility support module 208 for initializing the processor system 100 in a pre-boot environment (i.e., prior to executing an operating system boot process) in preparation for booting an operating system or otherwise starting an application program.

[0025] A shown in FIG. 3, an initialization process of a processor system (i.e., the processor system 100 of FIG. 1) may be performed by a processor such as the processor 102 of FIG. 1 based on, for example, the instructions and variables contained in the firmware image 202 of FIG. 2 (i.e., the processor boot instructions 204, the main firmware instructions or EFI 206, the compatibility support module 208 and the user/hardware variable space 210). It is known that the processor system initialization firmware image 202, as described above, may replace the BIOS in the processor system 100 and is responsible for initializing portions of the processor system 100 prior to executing an operating system boot process. Furthermore, the initialization process may begin following a reset event such as, for example, a system reset event or a power-on event. However, the initialization process may also begin as a result of other events that are not limited to reset events.

[0026] The processor 102 begins fetching and executing the processor boot instructions 204 at block 304. The processor boot instructions 204 cause the processor 102 to configure basic processor resources enabling it to further initialize portions of the processor system 100.

[0027] The processor 102 may begin to initialize portions of the processor system 100 by fetching and executing instructions from the EFI 206 (block 306). The EFI 206 is generally responsible for further initialization of the processor 102, performing system tests and initializing portions of the processor system 100 such as the display adapter 130, the input device 116, the network adapter 136, and the removable storage device drive 126. Furthermore, the EFI 206 establishes a hardware/software interface that allows an operating system or application program to run on the processor system 100 without knowing at least some of the specific hardware details of the processor system 100. The hardware/software interface and initializations performed by the EFI 206 provide support for newer and future operating systems and application programs. However, the EFI 206 may also enable support for legacy operating systems and application programs through the compatibility support module 208.

[0028] In the known initialization process of FIG. 3, the EFI 206 loads and executes the compatibility support module 208 (block 308) from the flash memory 110 on which the EFI 206 is stored. Furthermore, loading and executing the compatibility support module 208 (block 308) is an integral process of the EFI 206 that is always performed, regardless of whether the software boot target (i.e., the operating system or application program) requires legacy support. In this manner, the processor system 100 is always configured to support hardware/software interfaces that allow newer and legacy operating systems and application programs to communicate with the hardware comprising the processor system 100. Accordingly, the processor system 100 may load and execute a software boot target (block 312).

[0029] As stated above, using the known memory allocation configuration, the entire processor system initialization firmware image 202 is stored on a single non-volatile computer readable medium or memory device (i.e., the flash memory 110).

Turning now to FIG. 4, an example memory allocation configuration allows the firmware image 202 of FIG. 2 to be stored on at least two non-volatile memory devices. In the example memory allocation configuration, the boot instruction segment 204, the main instruction segment, or EFI 206, and the user/hardware variable space 210 may be stored on the flash memory 110, while the compatibility support module 208 may be stored on a separate, alternate non-volatile memory device.

[0030] FIG. 4 illustrates the example memory allocation configuration that supports a cross-memory implementation of memory allocation configuration. Unlike the known memory allocation configuration described in connection with FIG. 2, the memory allocation configuration disclosed herein allows a compatibility support module 208A, which may be substantially similar to the compatibility support module 208 described in greater detail in connection with FIG. 2, to be stored on an alternate non-volatile computer readable medium that is separate from the flash memory 110. For example, as shown in FIG. 4, the alternate non-volatile computer readable medium may be the mass storage device 122 (i.e., a hard disk drive) of FIG. 1. Furthermore, any alternate non-volatile computer readable medium that may be communicatively coupled to and read by a processor system (i.e., the processor system 100 of FIG. 1) may be used to store the compatibility support module 208A. For example, the removable storage medium 128 of FIG. 1 may be used to store the compatibility support module 208A. Storing the compatibility support module 208A on a separate non-volatile computer readable medium enables the flash memory 110 to be implemented by a smaller density flash memory device, thereby saving cost and space. Additionally, selective loading of the compatibility support module 208A based on need, saves boot time.

[0031] In one example, the compatibility support module 208A may be stored in a secure or protected area to prevent intentional or accidental tampering and/or corruption. As shown in the example of FIG. 4, the compatibility support module 208A is stored on a secure area or host protected area (HPA) 124, of the hard disk drive 122. As will be readily apparent by those ordinarily skilled in the art, the HPA 124 is a secure storage area that is protected by the EFI 206 in a manner such that a user is prohibited from accidentally or intentionally accessing its contents. This security may be implemented by requiring the processor system 100 to be in a special operating mode and/or requiring the user to provide a password. In other words, storing the compatibility support module 208A on the HPA 124 provides a level of security that prohibits a user or software process from accessing it and/or corrupting it.

[0032] Alternatively, the compatibility support module 208A may be stored on a protected system partition (not shown). The protected system partition may be created by the EFI 206 and, similar to the HPA 124, provides security against accidental or intentional accesses to information or data that is stored on it such as the compatibility support module 208A.

[0033] In some instances, the compatibility support module 208A may be stored on and provided by an externally accessible source. The externally accessible source may include a removable computer readable medium, such as a processor system utility disk (i.e., the removable storage media 128 of FIG. 1). Alternatively, the external source may include at least one of the processor systems 144 capable of communicating with the processor system 100 through the network 140, as shown in FIG. 1. Furthermore, prior to an operating system boot process the processor system 100 may selectively load and execute the compatibility support module 208A from

any of these external sources, or the processor system 100 may copy the compatibility support module 208A to a local computer readable medium such as the hard disk drive 122.

[0034] FIG. 5 is a flow diagram of an example initialization process 500 that may be executed by, for example, the processor system 100 of FIG. 1. The process 500 may be performed by a processor such as the processor 102 of FIG. 1 based on the instructions and variables comprising the example memory allocation configuration of FIG. 4 (i.e., the processor boot instructions 204, the main firmware instructions or EFI 206, the user/hardware variable space 210 stored on the flash memory 110 and the compatibility support module 208A stored on an alternate computer readable medium, such as the hard disk drive 122). As will become apparent in the following description, the disclosed process differs from the known method of FIG. 3 in that the EFI 206 enables legacy support in a conditional manner. In other words, the EFI 206 determines whether a software boot target requires legacy support and enables legacy support only if such support is required, thereby saving time in the initialization process when legacy support is not required. Also, because the compatibility support module 208A is stored on an alternate computer readable medium, the density of the flash memory 110 may be smaller thereby saving space and money.

[0035] The processor 102 begins to execute the processor boot instructions 204 at block 504. As described above, in connection with FIG. 3, the processor boot instructions 204 cause the processor 102 to enable and initialize basic processor resources, thereby enabling the processor 102 to further initialize portions of the processor system 100.

[0036] Instructions from the EFI 206 may then be fetched and executed at block 506. After further initialization of the processor 102, performing system tests, and

initializing portions of the processor system 100, the EFI 206 determines whether a software boot target requires legacy support (block 508). A software boot target may indicate that it requires legacy support by modifying a variable value such as, for example, a device path variable in the user/hardware variable space 210. The device path variable may be located in a boot options section of the user/hardware variable space 210 and a plurality of device path variables may exist if there are several software boot targets available to be loaded and executed by the processor system 100.

[0037] If the software boot target does not require legacy support, the software boot target is loaded and executed (block 510). Alternatively, if legacy support is required to support the software boot target, the processor 102 locates the compatibility support module 208A. The compatibility support module 208A may not be located on the same computer readable medium as the EFI 206, so the processor 102 locates the compatibility support module 208A on an alternate computer readable medium such as the hard disk drive 122 of Fig. 1. The location of the compatibility support module 208A may be determined based on a location variable in the boot options of the user/hardware variable space 210. The location variable may provide an identification of the alternate computer readable medium on which the compatibility support module 208A is stored. Alternatively, the processor system 102 may search a plurality of alternate computer readable media to determine where the compatibility support module 208A is stored by, for example, finding and matching a pre-determined identifier or signature pattern associated with compatibility support module 208A. The alternate computer readable media may include any computer readable media that is communicatively coupled to the processor in a fixed manner, such as the hard disk drive 122 and the ROM 108, or any

computer readable medium that is communicatively coupled to the processor system 100 in a removable manner, such as the removable storage medium 128. Additionally, the compatibility support module 208A may also be located on a remote processor system such as the processor system 144 of FIG. 1 communicatively coupled to the processor system 100 via the network 140.

[0038] Based on the location information, the compatibility support module 208A is loaded and executed at block 514. The compatibility support module 208A establishes a hardware/software interface and system parameters that provide legacy support, thereby enabling a legacy software boot target to run on the processor system 100. After establishing an appropriate hardware/software interface, the processor 102 loads and executes the software boot target (block 510).

[0039] Although certain methods, apparatus and articles of manufacture have been described herein, the scope of coverage of this patent is not limited thereto. To the contrary, this patent covers all methods, apparatus and articles of manufacture fairly falling within the scope of the appended claims either literally or under the doctrine of equivalents.